Genia Ortis *

DESIGN OF A TRAFFIC MICROSIMULATION IN UML

From the programming point of view traffic microsimulations consist of concurrent, interacting processes. Several representations of the same type (i.e. a car) are simulated over time. This structure highly corresponds to the concept of object oriented programming, which is therefore the first choice for the implementation of a traffic microsimulation. However, object oriented software has to be designed througly before the first line of program code is writen. Otherwise, with the expansion of the software and the increase of the number of developers a serious danger of incompatibilities and incorrect model behaviour arises. There exists a standard to describe object oriented models, the Unified Modelling Language UML. It can be used to describe a software from different points of view and therefore facilitates acquaintance and coordination processes and even code generation. In this paper I give an introduction into UML especially for developers of traffic microsimulations.

1. Introduction

Within the framework of the EU project CETRA (CEnter for TRAnsportation research) I had the opportunity to work half a year at the Institute for Transportation Networks at Zilina University. Since several years, the institute has been doing research in the field of simulation. One of the resulting products which is successfully commercialised is Villon, a software to simulate marshalling yards.

Several years ago the idea arose to use the experience, concepts and possibly some tools of railway and truck simulation for a microsimulation of car traffic. Some components of it are already designed or even implemented, e.g. parts of the graphic representation and the data structure of the network graph.

The development and implementation of this model is partly done by students, who are interested in a certain feature of it. I was invited to work at the microsimulation development as well. In order to get acquainted with the subject and to provide a comprehensive introduction for future developers of the model, I decided to create a structured top down description to form a powerful traffic simulation software suite.

If this description is carefully kept up-to-date, all work will be preserved and new modules can be designed to be compatible with the existing ones.

The UML (the Unified Modelling Language) is a widespread standard to describe object oriented software like a traffic simulation. In this article I show how to use the UML to describe a traffic simulation model. It is not my intention to provide a full description of the model but to illustrate with several examples how the UML is applied.

2. Components of a Microsimulation

2.1 Overview of existing traffic microsimulation tools

In order to get an overview of traffic microsimulations, I made a literature and internet review. A detailed description was found about the following models:

Traffic Microsimulations (not complete) Table 1

aaSIDRA	Akcelik & Associates Pty Ltd	http://www.aatraffic.com
AIMSUN 2	TSS – Transport Simulation Systems	http://www.tss-bcn.com/aimsun.html
CORSIM	Trafficware Corporation	http://www.trafficware.com
DRACULA	ITS, University of Leeds	http://www.its.leeds.ac.uk/projects/smartest
HUTSIM	Helsinki University of Technology	http://www.hut.fi/Units/Transportation/Research/hutsim.html
NEMIS	MIZAR Automazione S.p.a. Torino	–
SimTraffic	Trafficware Corporation	http://www.trafficware.com
SITRA B+	ONERA-CERT	http://www.cert.fr/fr/dcsd/CD/CDPUB/FINALITES/trafic.html
TRANSIMS	TSA-4, Los Alamos National Laboratory	http://www.transims.tsasa.lanl.gov
VISSIM	PTV	http://www.english.ptv.de/produkte/vissim.pl

* **Genia Ortis**
Department of Transportation Networks, Faculty of Management Science and Informatics, University of Žilina,
Tel.: +44-7952-641161, E-mail: genia.ortis@i-one.at

Object structure of the traffic microsimulation Table 2

basic item	representation in prototype	attributes	options for further development
Vehicle	car	kinematics, acceleration/deceleration rates, max. speed, driver aggressivity	split into other means of transport and pedestrians, characteristics of man/driver (i.e. aggressiveness)
network infrastructure	single net		several networks: public transport, car, pedestrian, bycicle, ...
	node	coordinates	
	edge	lane width and direction, allowed speed, length	
control infrastructure	traffic light	predefined signal control program	adaptive (to traffic conditions), coordinated (groups of traffic lights), optimisation of signal control program
	traffic panel	fixed content	variable message sign
traffic generation	single matrix	vehicles/hour foreach OD-pair	several matrices for several modes of transport
Time	timestamp		

As a result, objects, functions and results of the investigated traffic microsimulations were listed. This list forms the base for the UML model.

Table 2 shows the basic objects occurring in the traffic microsimulation. To ensure extensibility, known options for further development should be considered from the very beginning of the project.

This is the suggested structure of objects and attributes of the microsimulation to be designed. The initial UML-description of the software will be based on this structure.

It is necessary to define functions (i.e. car following, dynamic route guidance) and output parameters (i.e. emissions, travel time) of the microsimulation as well. They will be used to refine the initial UML structure.

3. Design of the Object Oriented Model in UML

UML is a standard defined by the Object Management Group (OMG) and is a widely spread toolset to define object oriented processes. "The Unified Modeling Language UML is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components." [OMG Unified Modeling Language Specification, Version 1.4, September 2001, see http://www.omg.org/cgi-bin/doc?formal/01-09-67, p. xix (Foreword)]

There exist many free and commercial applications, which support modeling in UML. The decision for one of them can be made according to the technical environment (operating system, programming language) and the requirements of the project (use as a project management tool as well, project size, team size). The following alternatives were considered for the traffic microsimulation prototype:

UML tools (not complete) Table 3

Free software:	"ArgoUML", a CASE-tool in/for Java
	"Ideogramic", a lightweight, intuitive UML-tool
	"UML modeller", a sourceforge project for Linux only
	"QuickUML"
Commercial software:	"Select Enterprise 6.2": a powerful project management tool,
	"Model Maker": a Module of the Delphi Enterprise edition
	"Poseidon": the commercial version of ArgoUML with additional OLE-functions
	"MagicDraw"
	"Rational Rose"
	"Together"

There exists an XML-based standard to exchange metadata like UML-models, named XMI. With XMI software specifications can be exchanged between different versions of the same software or even between different software tools.

UML documentation used to define the traffic microsimulation Table 4

Link	description	language
http://www.omg.org/cgi-bin/doc?formal/01-09-67	the formal UML specification	English
http://argouml.tigris.org/documentation/pdf/manual/argomanual.pdf	ArgoUML documentation	English
http://www.embarcadero.com/products/describe/umltutorials.asp	UML Tutorial	English
http://www.geocities.com/SiliconValley/Network/1582/uml-example.htm	UML by examples	English
http://www.rittershofer.de/info/uml/	UML Tutorial	German
http://ivs.cs.uni-magdeburg.de/%7Edumke/UML/index.htm	UML Tutorial	German

The internet provides a wide range of documentation about UML for different purposes and levels of knowledge. Table 4 contains a list of the UML descriptions and tutorials I referred to.

UML defines 9 types of diagrams, 4 describing the structure and 5 the behaviour of a software. Not all of them have to be used for the particular application. 5 different types of diagrams are sufficient to describe a traffic microsimulation. The diagrams can be drawn in any order, the sequence of the diagrams described in this paper is recommended to initially define the top-level description of the traffic microsimulation. Then, according to the progress of the software development, particular diagrams have to be refined down to the level of implementation.

3.1 Use Case Diagram

Starting point of UML modeling is the transfer of the user defined software behaviour (in human language) into a so called "Use Case Diagram". A Use Case Diagram contains the interactions (="Use Cases", drawn as ellipses) between Actors (User, Store) and the software.

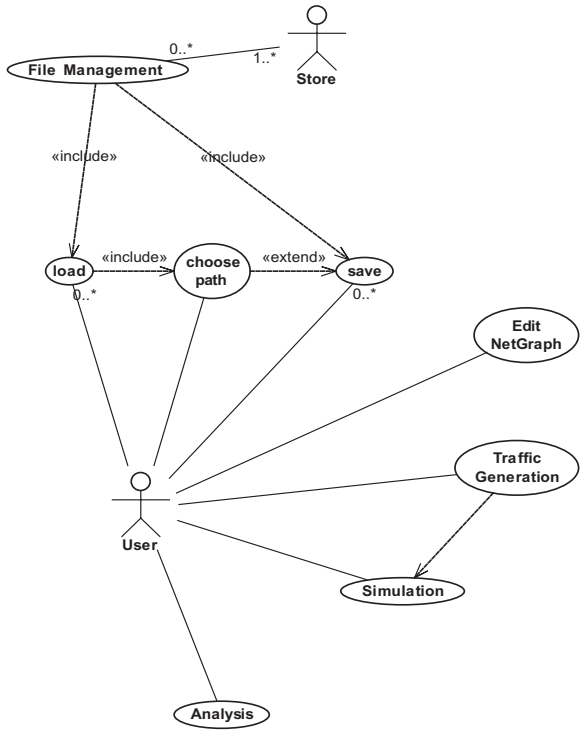

Fig. 1 Use Case Diagram

The association lines between actors and use cases represent communication paths. The association ends can be labeled with a multiplicity description.

Between use cases can exist different dependencies which are drawn as dashed arrows. A change to the target element may require a change to the source element in the dependency. The use case "load" "includes" the use case "choose path" because a filename and a path always have to be chosen. The use case "choose path" may "extend" the use case "save" subject to specific conditions ("save" vs. "save as").

3.2 Class Diagram

The object structure of the traffic microsimulation listed in table 2 is the base for the next step of UML modeling: The software structure is designed in a "Class Diagram". Classes, their methods and attributes as well as relations, (e. g, generalization) multiplicities and access paths between classes are visualized in the Class Diagram (see Fig. 2). Generalizations (i.e. inheritance) are visualized by a hollow diamond at the end of the more general object.

The result is a framework for the implementation. If there is a code-generator available for the chosen programming language, the skeletal source-code can be generated automatically.

3.3 Describing the Model Behaviour

To further specify the structure several behavioural diagrams are available in the UML. "Sequence Diagrams" and "Collaboration Diagrams" are used to describe the interaction between objects.

The Sequence Diagram has a timeline from top to bottom and shows sequential and concurrent flows within a process. The lifetime of an object instance is indicated by a bar along the dashed timeline. A solid arrow is a procedure call, while a dashed arrow is the return from a procedure.

The Collaboration Diagram focuses on the relationships among the objects and is a tool to define the procedures and functions in detail. The numbers correspond to the sequence of the procedure calls, nested numbers indicate nested procedure calls.

3.4 Describing the Behaviour of Single Objects

"State chart Diagrams" and "Activity Diagrams" focus on single objects and serve as a further specification of them.

Activity Diagrams are appropriate where procedures are triggered by the completion of previous procedures and define the sequence of actions within a procedure.

Alternatively in a State Chart Diagram states and transitions of an object triggered by asynchronous events during runtime are visualized.

4. Summary

The introduced technique is very well suitable to the development of a traffic microsimulation. Because of its object oriented

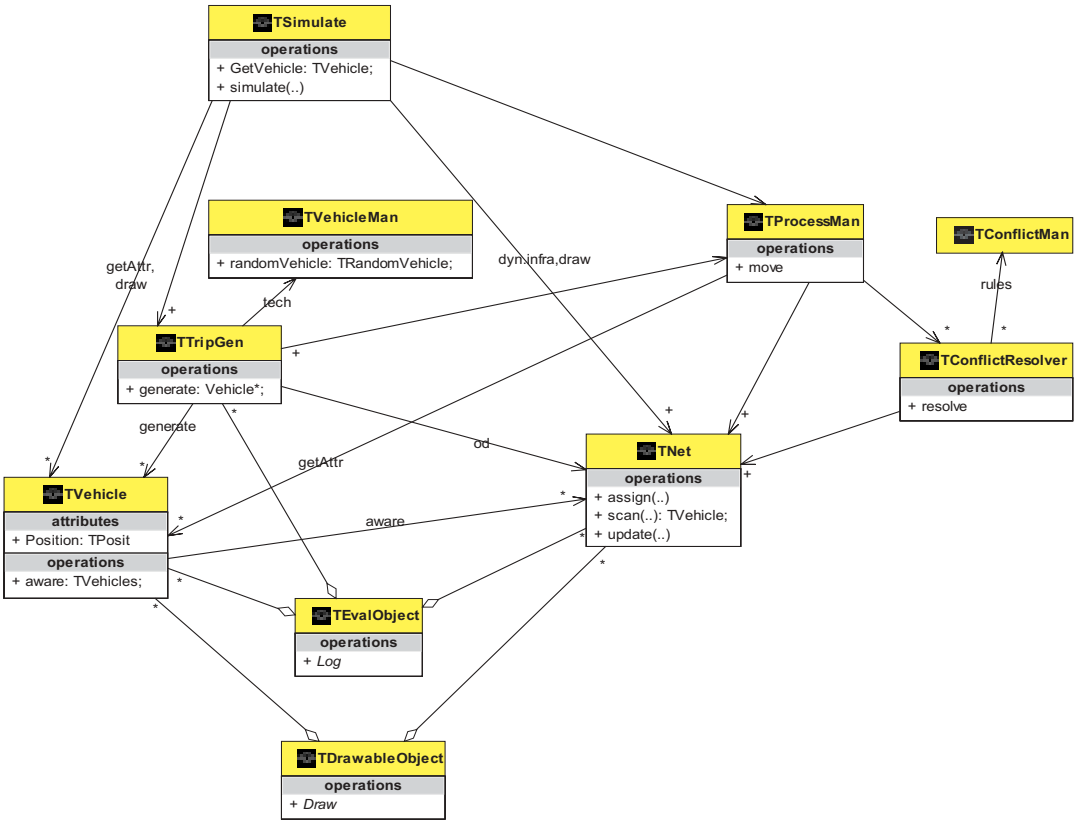

Fig. 2 Class Diagram of the Simulation Process

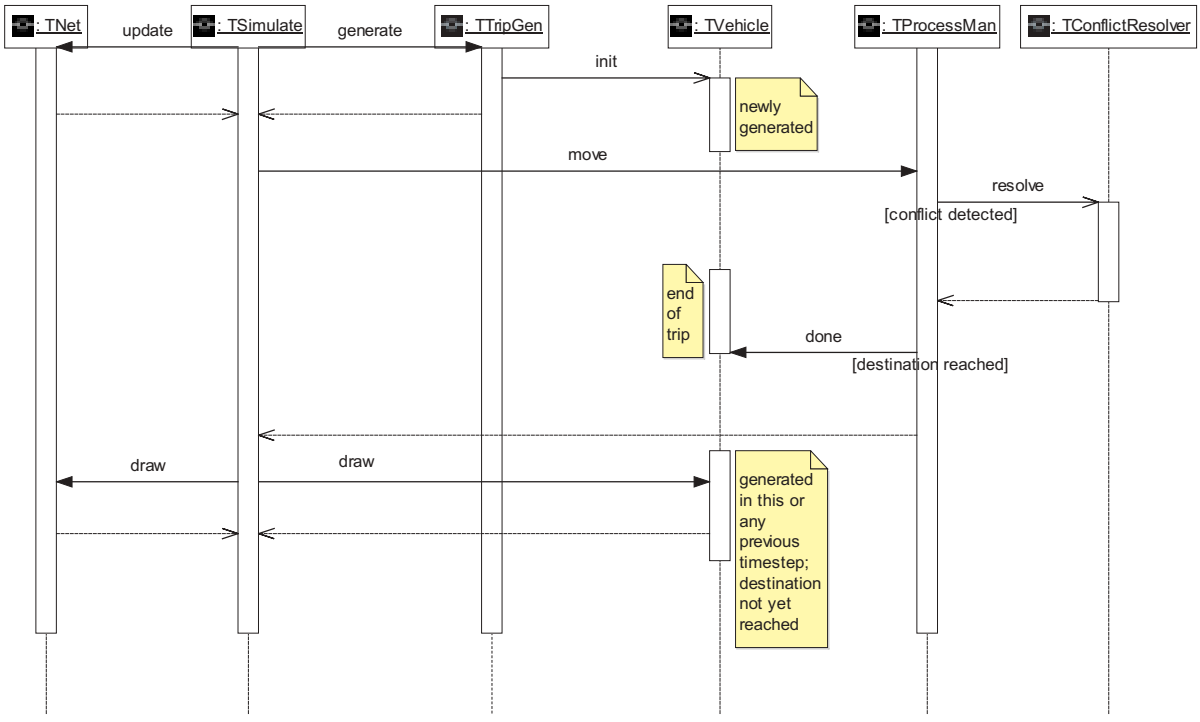

Fig. 3 Sequence Diagram of one simulation timestep

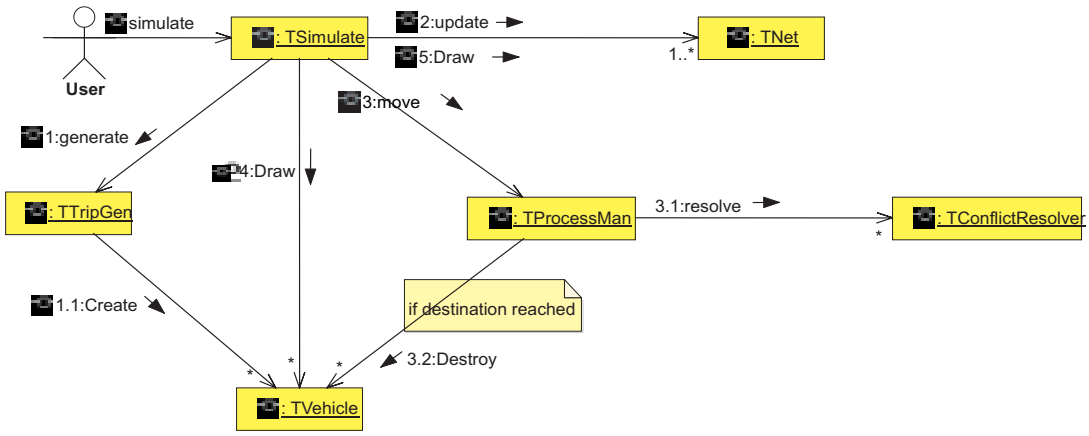

Fig. 4 Collaboration Diagram of one simulation timestep

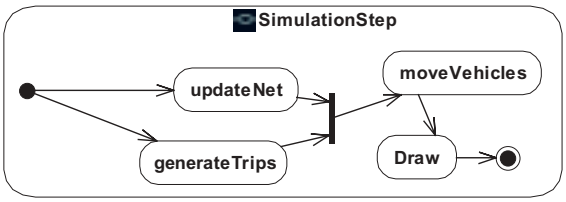

Figure 5 Activity Diagram of one simulation timestep

character the development of a simulation software requires a careful design process, before the first line of code can be written. The top-down approach with the UML enables an intuitive understanding of the necessary objects within the complex software structure. A consequent usage of the UML assures a consistent and flexible development-environment for a traffic simulation software suite. The reduced cost of software coordination and reengineering will many times pay the initial effort of learning and applying the UML.

Further information about the UML can be found in the documents listed in table 2.